

This document describes the operation  
of the FORTRAN language extensions  
for the VT11 Display Processor.

## **RT-11/FORTRAN Graphics Extensions User's Guide**

Order No. DEC-11-LFGMA-A-D

**SUPERSESION/UPDATE INFORMATION:**

This document requires the update document of  
the same name, Order No. DEC-11-LFGMA-A-DN1,  
published October 1977.

**OPERATING SYSTEM AND VERSION:**

RT-11 V03

**SOFTWARE VERSION:**

RT-11/FORTRAN Extensions V02

To order additional copies of this document, contact the Software Distribution  
Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation • maynard. massachusetts

First Printing, August 1976

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

Copyright © 1976 by Digital Equipment Corporation

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

|               |              |            |
|---------------|--------------|------------|
| DIGITAL       | DECsystem-10 | MASSBUS    |
| DEC           | DECTape      | OMNIBUS    |
| PDP           | DIBOL        | OS/8       |
| DECUS         | EDUSYSTEM    | PHA        |
| UNIBUS        | FLIP CHIP    | RSTS       |
| COMPUTER LABS | FOCAL        | RSX        |
| COMTEX        | INDAC        | TYPESET-8  |
| DDT           | LAB-8        | TYPESET-10 |
| DECCOMM       | DECSYSTEM-20 | TYPESET-11 |



## CONTENTS

|                                                                    | PAGE |
|--------------------------------------------------------------------|------|
| PREFACE                                                            | v    |
| CHAPTER 1      VT11 GRAPHICS SUPPORT                               | 1-1  |
| 1.1      INTRODUCTION                                              | 1-1  |
| 1.1.1      Hardware/Software Environment                           | 1-1  |
| 1.1.2      The VT11 Graphics Subsystem                             | 1-2  |
| 1.1.3      The Display File                                        | 1-3  |
| 1.2      VT11 DISPLAY PROCESSOR CONTROL ROUTINES                   | 1-3  |
| 1.2.1      Controlling the Display File                            | 1-8  |
| 1.2.2      Scaling the Screen Coordinates                          | 1-12 |
| 1.2.3      Positioning the Beam                                    | 1-13 |
| 1.2.4      Drawing Vectors                                         | 1-15 |
| 1.2.5      Displaying Text                                         | 1-16 |
| 1.2.6      Creating Subpictures                                    | 1-19 |
| 1.2.7      Handling Light Pen Interaction                          | 1-22 |
| 1.2.8      Using Graphic Arrays: Graphs and Figures                | 1-24 |
| 1.2.9      Using Timing Routines                                   | 1-27 |
| 1.2.10     Condensing, Storing, and Retrieving<br>the Display File | 1-29 |
| CHAPTER 2      BUILDING AND ALTERING THE GRAPHICS PACKAGE          | 2-1  |
| 2.1      BUILDING A LOAD MODULE                                    | 2-1  |
| 2.2      TECHNICAL DESCRIPTION OF DISPLAY FILE                     | 2-2  |

## FIGURES

|                                        |     |
|----------------------------------------|-----|
| FIGURE    1-1      VT11 Display Screen | 1-3 |
|----------------------------------------|-----|

## TABLES

|                                            |     |
|--------------------------------------------|-----|
| TABLE      1-1      VT-11 Display Routines | 1-4 |
|--------------------------------------------|-----|

1000

1010

1020

1030

1040

1050

1060

1070

1080

1090

1100

1110

1120

1130

1140

1150

1160

1170

1180

1190

1200

1210

1220



## PREFACE

This manual describes the use of the VT11 graphics extensions to FORTRAN/RT-11 that allow you to interact with the PDP-11 VT11 Graphics Display Subsystem.

VT11 support is provided as a set of FORTRAN-callable subroutines. Specifically, it is a set of library modules to be linked with your program. This library is distributed as part of the FORTRAN/RT-11 Extensions package. Most of the routines are designed to be as independent as possible from the actual structure of FORTRAN.

This manual describes the support for the VT11 Processor. This processor is an integral part of the GT44 and GT46 Graphical Display Systems. The user has complete use of the processor, which is controlled by the RT11 VT11 Handler (DEC-11-OVTMA-A-D). A majority of the routines implemented for the VT11 are system-independent and are available to users of other graphics systems.

All programs (or excerpts of programs when suitably completed) provided as examples in this manual may be run in either the single-job or foreground/background environment.

It is assumed that the user of the graphics extensions described in this manual is familiar with the FORTRAN language and has particular experience with FORTRAN under the RT-11 monitor.

### Documentation Conventions

The following list summarizes the documentation conventions used in describing the graphics calls in this manual.

| Convention                                                         | Meaning                                                                                                                                                                |
|--------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Square brackets [ ]                                                | Optional arguments are enclosed.                                                                                                                                       |
| FORTRAN variable names with upper-case letters (e.g., X11, I0, M2) | Standard default FORTRAN variable types whose value is returned by the routine being called; alternately, an array name.                                               |
| FORTRAN variable names with lower-case letters x11, i0, m2         | Standard default FORTRAN variable types whose value is to be supplied by the user. May be any valid arithmetic expression of that type (e.g., x is real i is integer). |
| [1[,i[,f[,t]]]]                                                    | The only exceptions to the above: special graphics parameters that are all integers (see Section 1.2.3).                                                               |

|        |                     |
|--------|---------------------|
| x-axis | The horizontal axis |
| y-axis | The vertical axis   |

#### NOTE

All floating-point constants supplied by the user as arguments in calls to the various modules should have a decimal point. If you do not strictly adhere to this convention in writing programs, the results of executing such programs are unpredictable.

The following list of manuals provides the necessary references for this document.

PDP-11 FORTRAN Language Reference Manual  
DEC-11-LFLRA-C-D

RT-11 System User's Guide  
DEC-11-ORGDA-A-D

RT-11 Advanced Programmer's Guide  
DEC-11-ORAPA-A-D

VT-11 Graphics Display Processor  
DEC-11-HVTGA-A-D

GT44 Users Guide  
EK-GT44-OP-001



## CHAPTER 1

### VT11 GRAPHICS SUPPORT

#### 1.1 INTRODUCTION

A comprehensive set of graphics extensions to FORTRAN/RT-11 has been implemented to provide support for the VT11 Display Processor. This chapter describes the use of the graphics subroutines in controlling the VT11. These routines are intended to be used in conjunction with the VT11 handler routines.

The graphics routines described in this chapter allow you to perform the following graphics functions by issuing simple CALL statements from your FORTRAN programs:

- . Display points, vectors, text, and graph data
- . Scale the display screen to any coordinate system
- . Control portions of the display independently by means of the subpicture facility
- . Facilitate light pen interaction
- . Display an entire array of data with one subroutine call

All necessary information on FORTRAN arithmetic, operations, functions, statements, and commands is included in the PDP-11 FORTRAN Language Reference Manual. This chapter describes the particular use of the calls that can be included in your FORTRAN programs to invoke the VT11 graphics routines. Section 1.2 summarizes these routines in a sequence of functional categories. Section 2.1 contains instructions on building a load module. Section 2.2 describes the internal structure of the display file. The software supplied to VT11 users is provided on DECTape, nine-track magnetic tape, floppy disk, or DECpack disk.

##### 1.1.1 Hardware/Software Environment

The hardware required for support of the VT11 graphics routines includes the following:

- . Any RT-11 system with at least 16K words of memory
- . VT11 display processor and screen

In addition, disk or another mass-storage device is usually needed to support the FORTRAN/RT-11 system and to facilitate development of user programs that utilize graphics capabilities. The TIMER and TIMR subroutines (see Section 1.2.9) also require access to a



## VT11 GRAPHICS SUPPORT

line-frequency clock. The amount of memory required to support the graphics system is dependent on your own programming requirements.

The graphics subroutines described below are generally intended to be used in conjunction with FORTRAN programs and to run under the RT-11 operating system.

### 1.1.2 The VT11 Graphics Subsystem

The VT11 Graphics Subsystem contains a display processor that is interfaced to the PDP-11 UNIBUS and is used primarily in the GT40/GT42/GT43 and GT44/GT46 series of PDP-11 graphics systems. The processor is controlled by the RT-11 VT11 Handler.

The VT11 display screen (Figure 1-1) provides a basic 9 1/4-inch by 9 1/4-inch (20.74-centimeter by 20.74-centimeter) viewing area. The screen can logically be treated as a coordinate grid with a horizontal x-axis and a vertical y-axis. There are 1024 logical positions on each axis, resulting in 1,048,576 individually addressable positions on the screen. You can address any of these positions by identifying the appropriate x and y coordinates associated with the desired position.

The viewing area capacity is 73 characters per line and 31 lines per screen.

Positions begin at (x=0, y=0) in the bottom left corner of the screen and extend to (x=1023, y=1023) at the top right corner of the screen. This is illustrated in Figure 1-1.

If you have a GT40 or GT42 system, your coordinate grid consists of 1024 positions on the x-axis (horizontal), but 768 positions on the y-axis (vertical). This results in 786,432 individually addressable positions on the screen.

If you define points or vectors that extend beyond the viewing area defined by the x and y coordinates, your display will be truncated or clipped at the edge of the screen.



## VT11 GRAPHICS SUPPORT

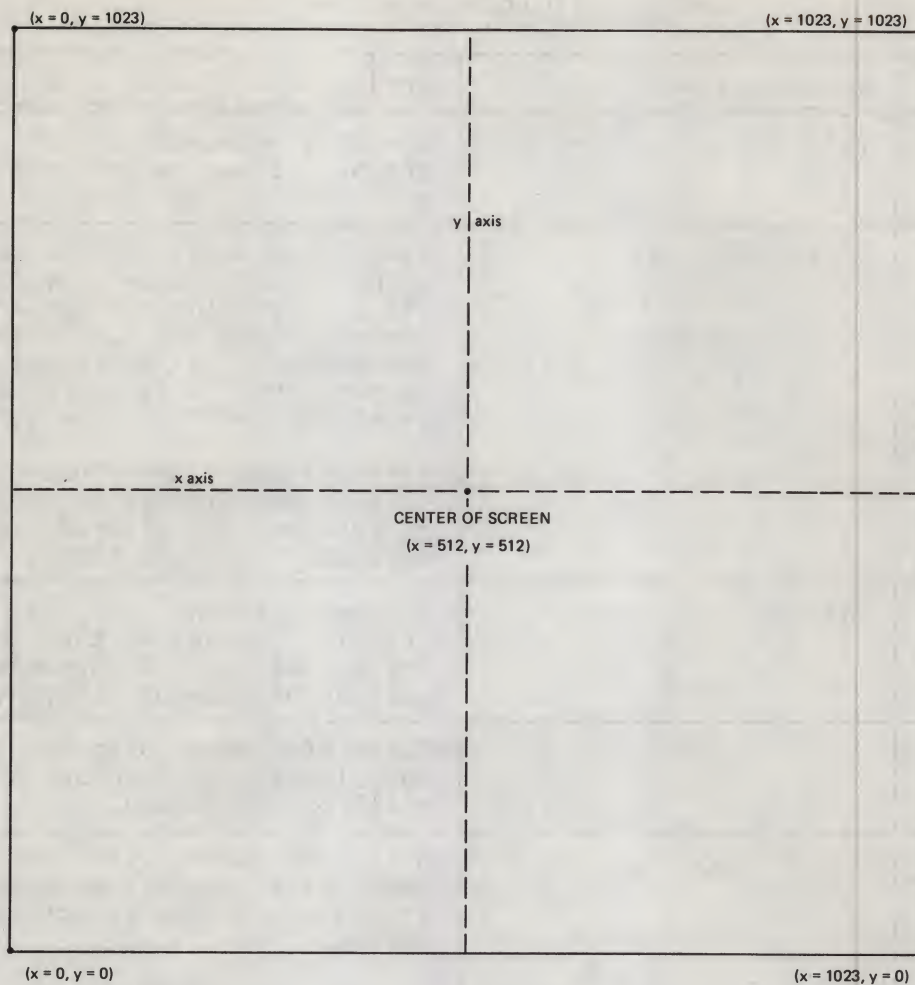


Figure 1-1 VT11 Display Screen

### 1.1.3 The Display File

When using the VT11 graphics routines in the RT-11 environment, you are allocated an area of memory for use as a display file and are given complete control over the use of this file. The instructions and data used to create graphic displays on the VT11 are contained in the display file. Almost any graphic display may be saved as a file on a mass-storage device such as disk or DECTape. The only exception is the storage of graph and figure arrays (see Section 1.2.8). Any file saved in this fashion may subsequently be restored, causing the original display to appear on the screen without reference to the FORTRAN program that was originally used to create the display.

### 1.2 VT11 DISPLAY PROCESSOR CONTROL ROUTINES

A variety of graphics control subroutines have been implemented to support the VT11 display processor. You can issue calls to these routines from FORTRAN programs under RT-11. Table 1-1 summarizes the names, argument lists, and effects of the graphics calls issued by FORTRAN programs for VT11 support.

VT11. GRAPHICS SUPPORT

Table 1-1  
VT11 Display Routines

| Call   | Argument List   | Effect                                                                                                                                                                                                                                                                                                                             |
|--------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AGET   | (A(i),Z)        | Unscales element i of the graphic array A and stores in Z.                                                                                                                                                                                                                                                                         |
| APNT   | (x,y[,l,i,f,t]) | Positions beam at absolute point represented by (x,y) after scaling, optionally changing the l, i, f, and t parameters; l is light pen sensitivity, i is intensity, f is flash, and t is line type.<br><br>If the x and y coordinates are outside the display screen, the coordinates are clipped (i.e., corrected to 0 and 1023). |
| APUT   | (A(i),b)        | Assigns element i of the graphic array A the scaled value of b. Dynamically changes display of array A.                                                                                                                                                                                                                            |
| BLNK   |                 | Turns off your display file, but leaves any scroller output still on the screen.                                                                                                                                                                                                                                                   |
| CMPRS  |                 | Acts like SAVE with no file name (see below), except that it leaves display intact on the screen.<br><br>CAUTION<br><br>The CALL CMPRS statement is illegal within a subpicture and causes a fatal error.                                                                                                                          |
| CONT   |                 | Restarts the display after a call to STOP.                                                                                                                                                                                                                                                                                         |
| DPTR   | (I)             | Returns in I the subscript of the display file array in which the next word of the display file will be stored.                                                                                                                                                                                                                    |
| DPYNOP | (n)             | Inserts n display no-ops into the display file, starting at the current position.                                                                                                                                                                                                                                                  |
| DPYWD  | (i,j)           | Puts data word i (16 bits) into the display file. An insert is performed only when j is zero, allowing you to put several words in at once before displaying the data.                                                                                                                                                             |



# VT11 GRAPHICS SUPPORT

Table 1-1 (Cont.)  
VT11 Display Routines

| Call  | Argument List    | Effect                                                                                                                                                                                                                                                  |
|-------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERAS  | [ (m) ]          | Erases the subpicture with tag m. If m is not included, this call erases the tracking object created by a call to TRAK.<br><br>CAUTION<br><br>The ERAS module acts as a no-op when an illegal tag is used.                                              |
| ESUB  |                  | Terminates the subpicture created by a call to SUBP with one argument.                                                                                                                                                                                  |
| FIGR  | (A,n[,l,i,f,t])  | Creates a figure using the first n elements of array A as x and y increment pairs, and optionally changes l, i, f, and t parameters.                                                                                                                    |
| FPUT  | (A(i),b)         | Assigns element i of graphic array A the scaled value of b and compensates the i+2nd element of A to leave following points of figure at same absolute location. Dynamically changes display of array A.                                                |
| FREE  |                  | Eliminates display file and clears the screen.                                                                                                                                                                                                          |
| INIT  | [ (IBUF,n) ]     | Initializes display processor to use the first n words of an integer array IBUF as a display file.                                                                                                                                                      |
| LPEN  | (M,N[,X,Y])      | Records a light pen hit (in M), the tag of a subpicture in which the hit occurred (in N), and, optionally, the x and y coordinates of the hit.                                                                                                          |
| LVECT | (x,y[,l,i,f,t])  | Draws a line segment from the current beam position (x0,y0) to the point (x0+x, y0+y) in long vector mode, optionally changing l, i, f and t parameters.                                                                                                |
| NMBR  | (n,var[,format]) | Creates a number subpicture with tag n, and displays the numeric value of the variable or expression in the optionally specified format. If format is not specified, the last format (specified) is effective. Initially the default format is 'F16.8'. |



# VT11 GRAPHICS SUPPORT

Table 1-1 (Cont.)  
VT11 Display Routines

| Call  | Argument List          | Effect                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NOSC  |                        | Eliminates scaling factor for subsequent graphics.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| OFF   | (t)                    | Turns off the subpicture with tag t.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| ON    | (t)                    | Turns on the subpicture with tag t.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| RDOT  | (x,y,[,i,l,f,t])       | Positions a beam originally at (x0,y0) to (x0+x,y0+y) scaled, optionally changing l, i, f, and t parameters.                                                                                                                                                                                                                                                                                                                                                              |
| RSTR  | ('[dev:]filnam[.ext]') | Restores the display of the file created by a call to SAVE. The default device is DK and the default extension is DPY.                                                                                                                                                                                                                                                                                                                                                    |
| SAVE  | ('[dev:]filnam[.ext]') | <p>Compacts the display file by eliminating references to erased subpictures and graphics arrays and if a file is specified, creates a copy of the graphic display on a file on DECTape or disk. Display may then be restored to the screen at any time by a call to RSTR. DK is the default device. DPY is the default extension.</p> <p style="text-align: center;">CAUTION</p> <p>The CALL SAVE statement is illegal within a subpicture and causes a fatal error.</p> |
| SCAL  | (x0,y0,x1,y1[,FX,FY])  | Scales the x axis to vary from x0 to x1 and the y axis to vary from y0 to y1, and optionally returns the X and Y scaling factors, in FX and FY, respectively.                                                                                                                                                                                                                                                                                                             |
| SCROL | (n,iy,[isw])           | Alters the RT-11 scroller parameters if you are currently in scroll mode.                                                                                                                                                                                                                                                                                                                                                                                                 |
| STAT  | (ns[,np])              | Enables or disables the italic mode, and optionally enables or disables visual intensification of light pen hit.                                                                                                                                                                                                                                                                                                                                                          |
| STOP  |                        | Stops display of the entire display buffer. Display may be restored by a call to CONT.                                                                                                                                                                                                                                                                                                                                                                                    |



# VT11 GRAPHICS SUPPORT

Table 1-1 (Cont.)  
VT11 Display Routines

| Call   | Argument List     | Effect                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SUBP   | (n1[,n2])         | When there is only one argument, begins definition of a subpicture with tag n1.<br><br>When there are two arguments, the new subpicture with tag n1 references an existing subpicture whose tag is n2.                                                                                                                                                                                                                       |
| TEXT   | (list)            | Outputs text to screen. List contains text to be printed, carriage return information, and information to convert text to the VT11 special shift-out character set.                                                                                                                                                                                                                                                          |
| TIMER  | (nz)              | Sets timer based on line frequency clock to a value of nz ticks, where each tick represents 1/60 second (for 60-Hz line current).                                                                                                                                                                                                                                                                                            |
| TIMR   | (IE)              | Returns the current value of the timer in variable IE.                                                                                                                                                                                                                                                                                                                                                                       |
| TRAK   | (X,Y)             | Puts a tracking object on the screen at the location (X,Y), the initial values of the variables, centers the object on any light pen hit within its area, and updates the values of X and Y to the new location.                                                                                                                                                                                                             |
| UNBLNK |                   | Turns on the display if previously turned off by a call to BLNK.                                                                                                                                                                                                                                                                                                                                                             |
| VECT   | (x,y[,l,i,f,t])   | Draws a line segment from the current beam position (x0,y0) to the point (x0+x, y0+y). Optionally changes l, i, f, and t parameters.                                                                                                                                                                                                                                                                                         |
| XGRA   | (y,A,n[,l,i,f,t]) | Plots the first n elements of the array A as a series of points, with the x positions determined by the scaled value of the elements of the array, and the y positions starting at the current beam location and incremented by y for each point. Optionally changes l, i, f, and t parameters.<br><br>If the specified x and y coordinates are outside the display screen, they are clipped (i.e., corrected to 0 or 1023). |



# VT11 GRAPHICS SUPPORT

Table 1-1 (Cont.)  
VT11 Display Routines

| Call | Argument List     | Effect                                                                                                                                                                                                                                                                                          |
|------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| YGRA | (x,A,n[,l,i,f,t]) | Plots the first n elements of the array A as a series of points, with the y positions determined by the scaled value of the elements of the array, and the x positions starting at the current beam location and incremented by x for each point. Optionally changes l, i, f, and t parameters. |

## NOTE

A user program that issues calls to the FORTRAN graphics routines should not issue the SYSLIB INTSET call or the monitor .PROTECT programmed request on the VT11 vectors. Multiple .PROTECT requests on the same vector cause a fatal system error. The FORTRAN Extensions software package protects the appropriate vectors before they are used.

If a user program uses more than one FORTRAN Extensions library package (VTLIB/LVLIB/LPSLIB), the monitor .INIT call should be issued after an abnormal termination of the program through a CTRL/C.

The following subsections describe in functional categories the graphics routines summarized in Table 1-1.

### 1.2.1 Controlling the Display File

(FREE, INIT, STOP, CONT, BLNK, UNBLNK, DPYNOP, DPYWD, and DPTR)

The VT11 display processor is similar to the central processor in that it retrieves instructions from memory. It is necessary to allocate the amount of memory to be used by the display processor. Once the display file has been created, any call to a graphics routine will put the display processor instructions into the display file. If the file becomes filled, a fatal DISPLAY BUFFER OVERFLOW error message is printed on the system console, and execution of the program is terminated.

## **FREE**

If you want to run a program that does not use the display after a certain point, and does need all available space for arrays and other



## VT11 GRAPHICS SUPPORT

data elements, any space currently allocated for the display file may be relinquished by issuing the following call:

CALL FREE

This routine may be called at any time and the screen will be cleared.

**INIT**

It is possible to erase and initialize (clear) the screen at any time by means of the following call:

CALL INIT

This routine clears the screen and initializes the display file.

The display file is originally set up by a call to INIT with two arguments. The first argument is the name of an integer array (\*2) space, and the second is the length in 16-bit words of the display file--for example:

```
DIMENSION IBUF(500)
CALL INIT(IBUF,400)
```

In this case, the first 400 words of the array IBUF is used as a display file and you are free to use the last 100 words of the array for other storage. You may reference this data by normal array subscripting. However, it is suggested that you be very careful when attempting to modify the display file through subscripted array references, because this may cause the display processor to generate an ILLEGAL MEMORY REFERENCE error if invalid data is entered.

Because it is necessary to call INIT in order to perform graphics work, any program that has called INIT will not return to the monitor immediately after execution or after an error. Instead, it will type the following message to the user:

TYPE <CR> TO EXIT

You will therefore be able to view your finished picture whether or not an error occurred, without having to implement an additional READ or PAUSE. You should not use the FORTRAN Library routine USEREX after a call to INIT; the graphics system uses this facility in the INIT logic.

**CONT  
STOP**

Two control routines have been implemented to allow the screen to be turned on and off. These are:

CALL CONT

CALL STOP

A call to STOP clears the screen, and a call to CONT restores to the screen the display that existed before the call to STOP. If a display



is stopped, then a call to STOP will have no effect. Similarly, if the display is running, a call to CONT will have no effect. These routines will halt the VT11 processor and allow the CPU to execute other parts of the program faster, because no interrupts from the VT11 will be serviced.

## BLNK UNBLNK

You may wish to have the display file turned off, but to keep the scroller and the tracking object on the screen; in that case, use:

```
CALL BLNK
```

```
CALL UNBLNK
```

These calls will blank and unblank your display. They can be used like STOP and CONT; unlike the STOP and CONT routines, however, BLNK and UNBLNK can keep the scroller output displayed.

## DPTR

You can access any word in the display file by means of array subscripting. The routine DPTR returns an integer value corresponding to the array element that is to be used for the next new piece of graphics code. This allows the user total control over the display file. The call is issued as follows:

```
CALL DPTR(I)
```

This routine will also allow you to see how much of the display buffer is actually in use. Note that if the last graphic data inserted was a long vector, and another long vector is to be inserted with the same intensity and other characteristics, then a mode word is not inserted. This means that the subscripted array element indexed by the value obtained by DPTR is at the x-coordinate of the vector. If the mode had changed, then the array element in question would be the mode word. Modes always change after subpictures and graphic arrays are specified. See the technical description of the file management for more complete details (Section 2.2).

You should not try to modify the display file by issuing DPTR calls unless you have a good understanding of the display file construction. Otherwise, unpredictable system behaviour may result.

## DYPNOP

You may want to insert some number of display no-ops in the display file, starting at the next array element (defined by I in the call DPTR(I)) in the display file for future modification by means of array subscripting. This can be accomplished with the following call:

```
CALL DYPNOP(n)
```



**DPYWD**

You may also want to insert your own display data into the file. This can be done by issuing the call:

```
CALL DPYWD(i,j)
```

where *i* is the data (16-bit for display instructions and 15-bit for display data with the MSB-bit set to 0) to be inserted, and *j* is a flag that indicates (if zero) that this inserted data is to be displayed. If *j* is non-zero, then it will not be displayed, thus allowing you to insert many pieces of data at once. Addresses may be inserted into the display file by means of the function IADRS, which returns the address of its argument. The argument should always be a variable and not an expression or a constant. You may have an array other than the display buffer that you would like to "link" into the display. For example, to insert a display jump to IA(25):

```
CALL DPYWD("160000,1)
CALL DPYWD(IADRS(IA(25)),0)
```

If you want to modify the display file by inserting a display jump, you should ensure that the intensity bit in the word at address IADRS(IA(25)) is set to the appropriate value. This is necessary because the current intensity (on/off) will remain in effect unless it is changed in that word.

**EXAMPLE**

Start a program by initializing the display file, set up display one, save a pointer to the current element in the display file, store two display no-ops in the display file, set up display two, and later store a display jump instruction to skip display one.

```

      DIMENSION IBUF(100)
C      INITIALIZE DISPLAY BUFFER
      CALL INIT(IBUF,100)
C      SET UP DISPLAY ONE
      ...
C      SAVE POINTER TO CURRENT ELEMENT IN DISPLAY BUFFER
C      INSERT TWO DISPLAY NO-OPS
      CALL DPTR(IP)
      CALL DPYNOP(2) !TO SKIP DISPLAY TWO IF NECESSARY
C      SET UP DISPLAY TWO
      ...
C      SKIP DISPLAY ONE
      CALL DPYWD("160000,1)
      CALL DPYWD(IADRS(IBUF(IP)),0)
      ...
```



1.2.2 Scaling the Screen Coordinates (SCAL and NOSC)**SCAL**

The VT11 display screen has a standard coordinate system with the lower left corner at (0,0) and the upper right corner at (1023,1023). Screens set up in rectangular format allow a maximum visible y-value of 768. You may want to work in another coordinate system; the SCAL routine allows you to change the standard VT11 coordinate system. To set the lower left corner of the screen at (x0,y0) and the upper right corner at (x1,y1), issue the following call:

```
CALL SCAL(x0,y0,x1,y1[,FX,FY])
```

Every call to a graphics routine made after the SCAL call will automatically assume that the lengths and coordinates are specified in terms of the scaling defined in the call to SCAL. It is possible to obtain the x and y direction scale factors by specifying values for the optional parameters, FX and FY, in the SCAL call. If a fifth parameter (FX) is included in the call, the x scale factor will be returned as the value of that parameter. If a sixth parameter is provided (FY), the y scale factor will also be returned.

At any time, you may override the current coordinate system by issuing another call to the SCAL routine. The new scale factors will only affect graphics calls following the new scale request.

Following is an example of a SCAL call:

```
CALL SCAL(0.,0.,200.,200.)
```

If a vector of length 200 in the x direction and 200 in the y direction is drawn from the point x=0,y=0, it will extend completely across the screen. Although the VT11 software does not facilitate the drawing of a single vector longer than the length of the screen, several vectors may be drawn consecutively to move the beam logically off the screen. If x0=x1 or y0=y1 in a SCAL call, the DIVISION BY 0 warning message will be displayed and the graphics specifications that follow will not work properly.

**NOTE**

In the call to SCAL, there is no requirement that x0 be less than x1 or that y0 be less than y1; pictures may therefore be drawn upside down or backwards, although text will always appear in left-to-right format.

**NOSC**

You may restore the VT11 standard coordinate system by issuing the following call:

```
CALL NOSC
```



## VT11 GRAPHICS SUPPORT

This turns the scaling routine off and reestablishes the default coordinate system. If no previous call to SCAL has been issued, a call to NOSC has no effect. You may alternatively restore the default coordinate system by establishing a unit scale:

```
CALL SCAL(0.,0.,1023.,1023.)
```

This call is much less efficient, because it invokes a scale with the x and y factors equal to one at every graphics call.

### EXAMPLE

Start a program by initializing the display file, scaling the screen, and later changing the scale and restarting the program.

```
C      LOCK MONITOR 'USR' MODULE IN CORE
      DIMENSION IBUF(500)
C      SET UP INITIAL WINDOW
      R=100.
1      CALL INIT(IBUF,500)
      CALL SCAL(-R,-R,R,R)
      . . .
C      RESCALE AND REDRAW
2      WRITE(5,90)
      READ(5,91) Z
      IF (Z .LE. 0) GO TO 2
      R=R/Z
      GO TO 1
90     FORMAT(' ENTER NEW RELATIVE SCALE')
91     FORMAT(F10.2)
      . . .
```

### 1.2.3 Positioning the Beam (APNT and RDOT)

#### APNT

To position the beam at an absolute point (x,y) on the screen, you should issue the following call:

```
CALL APNT(x,y[,l[,i[,f[,t]]]])
```

This will set the beam at the position specified by (x,y). The optional parameters l, i, f, and t are integer parameters whose values determine properties of the current and subsequent graphics functions. Because the call may include one, two, three, or all four of these parameters, subsequent call specifications will show these parameters as:

```
...[l, i, f, t]...
```

If the APNT call has been preceded by a call to SCAL, the x and y values will be scaled to their proper place on the screen. The scaled x or y may not position the beam at a negative value or a value exceeding 1023; x and y must therefore be within the range of x and y, as defined in any previous SCAL call.



## VT11 GRAPHICS SUPPORT

The following list summarizes the effects of the APNT parameters.

| Parameter | Effect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| l         | Light pen interaction. If a positive value for l is included in the call, then current and subsequent graphic output will be light pen sensitive (i.e., will cause a light pen interrupt when pointed to on the screen). If l is zero or omitted from the call, there is no change in the parameter from its previous setting. If l is negative, subsequent graphics will not be light pen sensitive.                                                                                                                                                                     |
| i         | Intensity. Points, vectors and text may be displayed on the screen in any of eight intensities. If a positive value (1 through 8) for i is included in the call, the intensity will be set to that value; i=8 is the brightest screen intensity. If i is zero or omitted from the call, then no change is made from the previous setting. If i has a negative value, the current graphics output (except text) is invisible. Additionally, if the negative value is between -8 and -1, the intensity is changed to the absolute value of i for subsequent graphics calls. |
| f         | Flash. If a positive value for f is included in the call, then the blink mode will be enabled, starting with the current graphic output. If f is zero or omitted from the call, no change is made in the blink mode. A negative value for f will disable the blink mode.                                                                                                                                                                                                                                                                                                  |
| t         | Type of line. If a t value of 1, 2, 3, or 4 is specified, the line type will be enabled to solid, long-dashed, short-dashed, or dot-dashed lines, respectively. If t is greater than 4 or non-positive, no change in the line type will be made. If t is omitted from the call, then no change is made in the current setting.                                                                                                                                                                                                                                            |

These parameters may also be changed in calls to RDOT, VECT, and LVECT. You may change the parameters without affecting the display by issuing either of the following:

- . RDOT call with the x and y arguments equal to zero, and a negative intensity
- . VECT call with the x and y arguments equal to zero (see Section 1.2.4).

### NOTE

A change in these parameters will have no effect on graphics previously drawn.

At the start of the display file, after a call to INIT, the default conditions are the following:

- . no scaling
- . beam at lower left corner



- . light pen interaction is disabled
- . intensity is set to 5
- . flash is turned off
- . solid line type is enabled

**RDOT**

You may insert dots on the screen relative to the current beam position by issuing:

```
CALL RDOT(x,y[,l,i,f,t])
```

This will set a beam originally at (x0,y0) to the position (x0+x,y0+y). As in the case of APNT, any current scaling will be performed automatically.

#### 1.2.4 Drawing Vectors (VECT and LVECT)

**VECT**

Vectors may be drawn by means of the following call:

```
CALL VECT(x,y[,l,i,f,t])
```

This routine will cause a line segment to be drawn from the current beam position, (x0,y0), to the point (X0+x,Y0+y). The absolute values of x and y must be less than 1024 after scaling. If x and y both have a zero value, then no output will be visible on the screen, regardless of the value of i.

**LVECT**

If it is possible to draw the specified vector as a short vector, then the VECT routine will draw it in this way. For this reason, you can instead use:

```
CALL LVECT(x,y[,l,i,f,t])
```

This routine will always cause a long vector to be drawn.

**EXAMPLE**

```
C    PROGRAM TO DEMONSTRATE APNT, RDOT, AND VECT
      DIMENSION IBUF(500)
      CALL INIT(IBUF,500)
      CALL APNT(200.,500.,0,-5,1)
C    THE BEAM IS POSITIONED INVISIBLY AT (200,500)
C    WITH THE FLASH MODE ENABLED.
      CALL VECT(0.,200.,0,0,0,1)
```

## VT11 GRAPHICS SUPPORT

```

C      A VERTICAL VECTOR (SOLID LINE) 200 UNITS LONG
C      IS DRAWN AND IT FLASHES BECAUSE OF THE
C      CALL TO APNT.
C      CALL RDOT(50.,0.,0,4,-1,3)
C      THE BEAM IS POSITIONED 50 UNITS TO THE
C      RIGHT OF THE TOP OF THE VECTOR, INTENSITY 4,
C      FLASH IS DISABLED, AND LINE TYPE IS SET
C      TO SHORT DASH.
C      CALL SCAL(-50.,-50.,50.,50.)
C      THE SCREEN IS SCALED.
C      CALL VECT(0.,-20.,1)
C      A VECTOR IS DRAWN APPROXIMATELY ONE QUARTER
C      DOWN THE SCREEN AND IS LIGHT PEN SENSITIVE
C      AND SHORT DASHED
C      CALL EXIT
C      END

```

The following example is an excerpt from a graphics program. It scales the screen for a window -50 to +50 in both directions, draws a diamond centered on the window, and finally displays a visible dot in the center of the diamond.

```

...
CALL SCAL(-50.,-50.,50.,50.)
CALL APNT(0.,25.,-1,-5,-1)
CALL VECT(-25.,-25.,0,0,0,1)
CALL VECT(25.,-25.)
CALL VECT(25.,25.)
CALL VECT(-25.,25.)
CALL RDOT(0.,-25.)
...

```

Note that the last line could be written:

```
CALL APNT(0.,0.)
```

### 1.2.5 Displaying Text (TEXT, SCROL, and STAT)

#### TEXT

Textual and carriage-control information may easily be displayed on the VT11 screen. The user causes text to be displayed by issuing the following:

```
CALL TEXT (<arglist>)
```

where arglist may contain any of the following elements separated by commas

| Element     | Effect                                                 |
|-------------|--------------------------------------------------------|
| String data | Text to be displayed on screen (ending in a null byte) |
| [+]m        | Carriage return and m line feeds                       |
| 0           | Carriage return and no line feed                       |
| -m          | Converts following text to shift-out characters.       |



# VT11 GRAPHICS SUPPORT

A carriage return causes the display of the next line of text, starting at the left edge of the screen; the value of m must be less than 64.

Normal text consists of any printable character with an ASCII value greater than or equal to 40 (octal). Shift-out characters are the special character set, consisting of 31 symbols and Greek letters, within the display processor. The following characters will be converted to shift-out characters when included in a string following -m:

@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^

The example included below provides a list of available shift-out characters. You should not include any character other than those listed above within the shift-out text. The letter O within shift-out text will be converted to a blank space on the display screen.

The location of the text printed may be controlled by positioning the beam with an invisible dot before the TEXT call. The lower left corner of the first character printed will start at this point. A single character fits in a 6-by-8 dot matrix on the VT11 screen.

## EXAMPLE:

```
C   THIS PROGRAM DEMONSTRATES THE USE OF
C   TEXT AND THE SPECIAL CHARACTER SET
    DIMENSION IBUF(500)
    CALL INIT(IBUF,500)
    CALL APNT(100.,600.,0,-5)
    CALL TEXT('ASCII(10) 64 65 66 67 68 69 70 71 72 73 74')
    CALL APNT(100.,550.,0,-5)
    CALL TEXT('NORMAL      @ A B C D E F G H I J')
    CALL APNT(100.,500.,0,-5)
    CALL TEXT('SHIFT-OUT  ', -1, '@00A00B00C00D00E00F00G00H00I00J')
    CALL APNT(100.,400.,0,-5)
    CALL TEXT('ASCII(10)   75 76 77 78 79 80 81 82 83 84 85')
    CALL APNT(100.,350.,0,-5)
    CALL TEXT('NORMAL      K L M N O P Q R S T U')
    CALL APNT(100.,300.,0,-5)
    CALL TEXT('SHIFT-OUT  ', -1, 'K00L00M00N00000P00Q00R00S00T00U')
    CALL APNT(100.,200.,0,-5)
    CALL TEXT('ASCII(10)   86 87 88 89 90 91 92 93 94 95')
    CALL APNT(100.,150.,0,-5)
    CALL TEXT('NORMAL      V W X Y Z [ ] ')
    CALL APNT(100.,100.,0,-5)
    CALL TEXT('SHIFT-OUT  ', -1, 'V00W00X00Y00Z00[00o0]0000')
    END
```

|           |    |    |    |    |    |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|----|----|----|----|----|
| ASCII(10) | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 |
| NORMAL    | Q  | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  |
| SHIFT-OUT | h  | a  | b  | c  | d  | e  | f  | g  | h  | i  | j  |

|           |    |    |    |    |    |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|----|----|----|----|----|
| ASCII(10) | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 |
| NORMAL    | K  | L  | M  | N  | O  | P  | Q  | R  | S  | T  | U  |
| SHIFT-OUT | k  | l  | m  | n  | o  | p  | q  | r  | s  | t  | u  |

|           |    |    |    |    |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|----|----|----|----|
| ASCII(10) | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| NORMAL    | V  | W  | X  | Y  | Z  | [  | \  | ]  | ^  | _  |
| SHIFT-OUT | v  | w  | x  | y  | z  | [  | \  | ]  | ^  | _  |

String data must end with a null byte. Compile-time string constants satisfy this requirement, and you must remember this requirement if you are generating your own string data in numeric arrays.

#### NOTE

To change any of the parameters, *l*, *i*, *f*, or *t* for textual output, the call to TEXT must be preceded by:

```
CALL RDOT(0.,0.,l,-i,f,t)
```

Also, text material is always visible; that is, one can not write "invisible words" on the screen.

## STAT

Italics may be enabled by a call to the STAT routine that allows you to disable or enable the italic mode, and to disable or enable the intensification that occurs at a light pen hit. The call is of the form:

```
CALL STAT(ns[,np])
```



## VT11 GRAPHICS SUPPORT

The following list summarizes the effects of the STAT parameters.

| Parameter | Effect                                                                                                                                                                                                                                                                                                                           |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ns        | Font. If a positive value for ns is included in the call, then the italic font is disabled. If ns is zero, then there is no change from the previous setting. If ns is negative, then italic mode is enabled.                                                                                                                    |
| np        | Intensification of light pen hit. If a positive value for np is included in the call, then the point of a light pen hit will be raised to intensity level 8. If np is zero or omitted from the call, there is no change from the previous setting. If np is negative, the point of a light pen hit will not change in intensity. |

The default conditions at the start of the display file cause italic mode to be disabled and light pen intensification to be enabled. Do not confuse intensification of a light pen hit with interaction of a light pen (see the Section 1.2.7).

### SCROL

You can dynamically change the SCROLLER parameters in the monitor by calling the SCROL routine with two or three parameters:

```
CALL SCROL(n,iy[,isw])
```

The n parameter changes the number of lines of scroller code. The scrolling buffer can never be made larger than it already is. The iy parameter causes the lower left corner of the first character to be at the given unscaled-y coordinate. If parameter values of -1 are specified, the monitor default values will be reestablished. A value of zero means no change. The optional isw parameter may be included to set the intensity level of the scrolling text; this level must be in range 1 through 8 inclusive. Scrolled data cannot be blanked from the screen.

#### 1.2.6 Creating Subpictures (SUBP, ESUB, ON, OFF, ERAS, and NMBR)

### SUBP ESUB

A series of graphics calls may be grouped together and defined as a unit called a subpicture. The subpicture is assigned a name or tag in the SUBP call:

```
CALL SUBP(n1)
```



## VT11 GRAPHICS SUPPORT

By referencing this tag, you can issue graphics calls to:

- . display an instance of the subpicture
- . turn the subpicture on and off
- . erase the subpicture
- . test for light pen hits in the subpicture

For every call to SUBP with one argument, there should be a corresponding call to the ESUB routine after it:

CALL ESUB

If a SUBP call does not have a corresponding ESUB call associated with it, the graphics display will not appear on the screen until the ESUB call is issued.

Each subpicture (graph and number) has a unique tag associated with it. The n1 parameter is a positive integer less than 32768. A subpicture may alternatively be a display of a previous subpicture. To display a subpicture at any point on the screen, position the beam at the desired point and then issue a call to SUBP with two arguments. The form of the call is:

CALL SUBP(n1,n2)

where n2 is the tag of the subpicture being referenced and n1 is the tag of the new subpicture being created. This call creates a jump in the display buffer back to the display file code for the original subpicture. A call to SUBP with two arguments should not have a corresponding call to ESUB associated with it.

### EXAMPLE

Define a subpicture that draws a horizontal resistor and then reference it in another subpicture.

```

        DIMENSION IBUF(500)
        CALL INIT(IBUF,500)
        CALL APNT(500.,100.,0,-5)
        CALL SUBP(1)
        CALL VECT(20.,0.,0,5,0,1)
        CALL VECT(5.,10.)
        DO 1 I=1,2
        CALL VECT(10.,-20.)
1      CALL VECT(10.,20.)
        CALL VECT(10.,-20.)
        CALL VECT(5.,10.)
        CALL VECT(20.,0.)
        CALL ESUB
C      THE FIRST SUBPICTURE IS NOW COMPLETED
        CALL APNT(200.,400.,0,-5)
        CALL SUBP(2,1)
        CALL EXIT
        END
```

Any subpicture to be referenced should not contain any calls to APNT; graphic data should be relative, not absolute, or the subpicture will not be completely movable.

Note that subpictures may contain calls to other subpictures. The subpicture call depth limit is ten. A nested subpicture is created by



a call to SUBP with one argument, followed by a second call to SUBP before any calls to ESUB are issued. There can be as many as ten calls to SUBP before a call to ESUB, including the first call to SUBP. The first call to ESUB encountered in the code will end the last subpicture created. No graphics within the nested subpicture will appear on the display screen until one call to ESUB has been executed for each call to SUBP that contains only one tag. Nested subpictures are useful in applications where there are several component graphics that will be frequently referenced or tested for light pen hits as a group and individually. If there are more than ten subpictures nested, an ILLEGAL SUBPICTURE NESTING error message will be printed.

**ON  
OFF**

Subpictures may be turned on and off individually by means of the following calls:

CALL ON(n)

CALL OFF(n)

where n is the tag of the subpicture. When a subpicture is turned off, there will be two effects on the display:

1. The graphics calls included in the subpicture will not appear on the screen.
2. Any relative graphics call after the subpicture will appear relative to the beam position before the subpicture instead of after the subpicture.

Any subpicture to be turned off should therefore be followed by a call to APNT unless it is desirable to have the graphics calls moved. The code for the graphics remains in the display file, and a subpicture may be copied while it is turned off. Once a subpicture has been turned off, it may be turned on by issuing a call to ON. Turning on a subpicture that is already on or turning off a subpicture that is off has no effect. You may turn off a subpicture before it has completely been defined--that is, between the SUBP and ESUB calls that define it.

**ERAS**

When a subpicture is no longer needed, it may be erased by means of the ERAS routine. The call is issued as follows:

CALL ERAS(n)

where n is the tag of the subpicture. The difference between erasing a subpicture and turning it off is that erasing turns off the subpicture and also eliminates any reference to the code in the display file. This has the effect of freeing up the corresponding tags for subsequent use. Erasing a subpicture does not recapture that portion of the display file that was used for the subpicture; however once subpictures have been erased, the display file may be condensed by a call to SAVE or CMPRS (see Section 1.2.10).



**NMBR**

You can place numeric data on the screen using E, I, F, or O format with a maximum field width of 16. The data is defined in a special number subpicture that has a tag like regular subpictures, but can be updated in an odometer fashion.

The subroutine requires two or three arguments:

```
CALL NMBR(n,var[,format])
```

where n is the tag, which must be unique in the program, and var is the data to be displayed and is an expression of the type given in the format specified. If the format parameter is not included in the call, then the last format supplied is used. Initially, the default is F16.8. The format is specified as a string, ending in a null byte, in the following way:

```
CALL NMBR(n,X(i),'F8.2')
```

Any additional references to a number subpicture with the tag of an existing number subpicture will cause the referenced subpicture to be updated. You may alternatively use the DECODE facility with the TEXT routine to accomplish a similar effect.

The location of the number displayed may be controlled by positioning the beam with an invisible dot before the call to NMBR. The lower left corner of the first number printed will start at this point.

#### 1.2.7 Handling Light Pen Interaction (LPEN and TRAK)

**LPEN**

If any graphics on the screen have been made light pen sensitive, you can perform the following operations on these graphics by using the LPEN routine:

- . test for light pen hits
- . determine the x- and y-coordinates of the light pen hit
- . determine in which (if any) subpicture the hit occurred

The LPEN call is issued as follows:

```
CALL LPEN(M,N[,X,Y])
```

where M is a flag that is set equal to zero if no hit has occurred, and equal to one after a hit has occurred. If M equals one, then the value of N is the tag of the subpicture in which the hit occurred. If the hit occurred other than in a subpicture, N will be set to zero. If values for parameters X and Y are specified, the scaled x- and y-coordinate values of the hit will be stored in these variables.



**TRAK**

To allow you to utilize the light pen capability of the display processor to its fullest advantage, the graphics system provides for optional tracking with the light pen. If X and Y are variables, not constants, then:

CALL TRAK(X,Y)

displays a diamond-shaped object on the screen at scaled coordinate positions (X,Y); this object will react to light pen hits in such a way as to always center itself on the last light pen hit on the object. The tracking object will only respond to hits within its area. It can be seen by running a simple program. The initial position of the tracking object should be determined by an assignment statement to X and Y before the call to TRAK. At each light pen hit, X and Y are updated to the new center of the tracking object for program use.

**CAUTION**

Because X and Y are updated asynchronously with the CPU, the Optimizing Code of the FORTRAN Compiler may not have the same effect on these variables as a non-optimized version of the same program. Under circumstances such as global subexpression elimination (e.g., variable in conditional within a DO loop), the optimized code may not respect the asynchronous property of these variables, and unpredictable results may occur. The solution to this problem is to EQUIVALENCE any asynchronous variables to any other name, because EQUIVALENCED variables are not optimized.

The diamond-shaped object cannot be snapped off the screen by a fast motion of the light pen; upon reaching the edge, it will reposition itself near the center of the screen.

**NOTE**

On a rectangular screen, the initial values of X and Y (scaled to between 0 and 1023) may put the tracking object off the top of the screen.

A call to ERAS with no tag specified causes the tracking object to be removed from the screen:

CALL ERAS



### 1.2.8 Using Graphic Arrays: Graphs and Figures (YGRA, XGRA, AGET, APUT, FIGR, and FPUT)

FORTRAN/RT-11 with graphics support provides three graphic array calls (YGRA, XGRA, and FIGR) that allow you to display an entire array by specifying a single graphics call. Graphics created by these calls may be changed while they are being displayed, by issuing calls to APUT, AGET, and FPUT. Because the graphics created may be dynamically changed, you should be extremely careful when using these calls. After a YGRA, XGRA, or FIGR call has been made, you must issue a call to INIT or must erase the subpicture containing the array before your array is available for general use. In order to maintain calling-sequence compatibility with BASIC/GT, certain array subscript checking was not possible; hence, the user must be very careful in graph array references.

#### YGRA

It is possible to plot a graph of points by issuing a series of APNT calls. In the case of y-graph data, the x coordinates are often evenly spaced across the screen. In this situation, you can place the y-coordinate data in an array and display directly from the array by means of the following call:

```
CALL YGRA(x,A,n[,l,i,f,t])
```

where x is the increment in the X direction and A is the name of a 1-dimensional real array whose first n entries will be plotted. The X position will be incremented by x, starting at the current beam position. Note that the y-array data must be stored as absolute coordinates.

#### XGRA

Similarly, you can display a graph of X-data by issuing the following:

```
CALL XGRA(y,B,m[,l,i,f,t])
```

where y is the increment in the y direction and B is the name of a 1-dimensional real array whose first m words will be plotted.

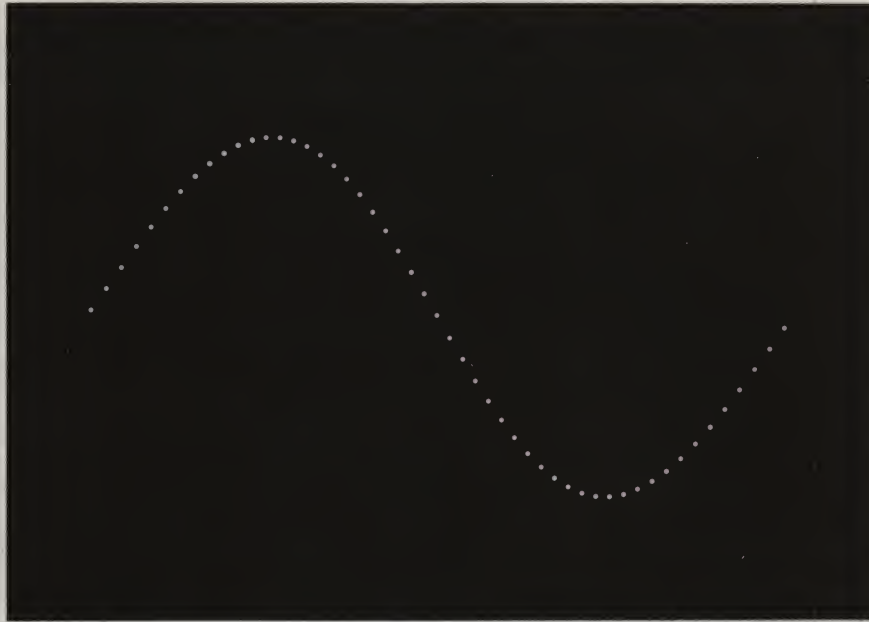
#### EXAMPLE

Draw a sine wave across the entire screen using 51 points.

```

      DIMENSION R(51),IBUF(50)
      DATA PI/3.1415926535/
      DO 1 I=1,51
1      R(I)=200.+200.*SIN(PI*(I-1)/25.)
      CALL INIT(IBUF,50)
      CALL YGRA(20.,R,51)
      STOP
      END
```





**AGET  
APUT**

If you are extremely careful, you can access graph data by means of array subscripts. It is far simpler to issue the following calls:

```
CALL AGET(A(i),Z)
```

```
CALL APUT(A(i),b)
```

where A is the name of a graph data array, i is a valid subscript, and Z is a scalar variable (not a scalar whose name is also a graph data array). AGET unscales the A(i) and returns a value in Z. APUT scales the value of b and stores the correct value in A(i). The display will be immediately updated.

**FIGR**

Further graphics capabilities are available, particularly regarding the motion of objects. You may draw a figure from the current beam position as follows

```
CALL FIGR(A,n[,l,i,f,t])
```

The figure is drawn from an array (A) of pairs of vectors. (A(1),A(2)) is the first x,y-pair, A(3),A(4) is the second x,y-pair, and so on. The value of n must be even. These arrays may be accessed by the AGET and APUT routines as described above. A call to APUT will change the appropriate vector and will cause all subsequent vectors to be displaced.

## EXAMPLE

This example uses an "invisible" figure to move a graphics display.

```

      DIMENSION IBUF(100),R(2)
      . . .
      CALL FIGR(R,2,0,-5)
      . . .
C     MOVE THE GRAPHICS
      CALL APUT(R(1),X)
      CALL APUT(R(2),Y)
      . . .

```

**FPUT**

Alternatively, if you want to alter only one point in a displayed figure, you can issue a call to FPUT. The form of the call is:

```
CALL FPUT(A(i),b)
```

This call will change the *i*th element in array *A* to the value of *b* scaled and will then compensate *A(i+2)* to keep the following points in the figure in the same location. The call:

```
CALL FPUT(A(3),100)
```

is equivalent in effect to the following series of AGET and APUT calls:

```

CALL AGET(A(3),Y)
CALL AGET(A(5),Z)
CALL APUT(A(3),100.)
CALL APUT(A(5),(Z+Y-100.))

```

If there is no *A(i+2)*, a call to FPUT(*a(i)*,*b*) is equivalent to a call to APUT(*a(i)*,*b*).

## EXAMPLE

This example demonstrates the use of the FIGR, APUT, and FPUT instructions.

```

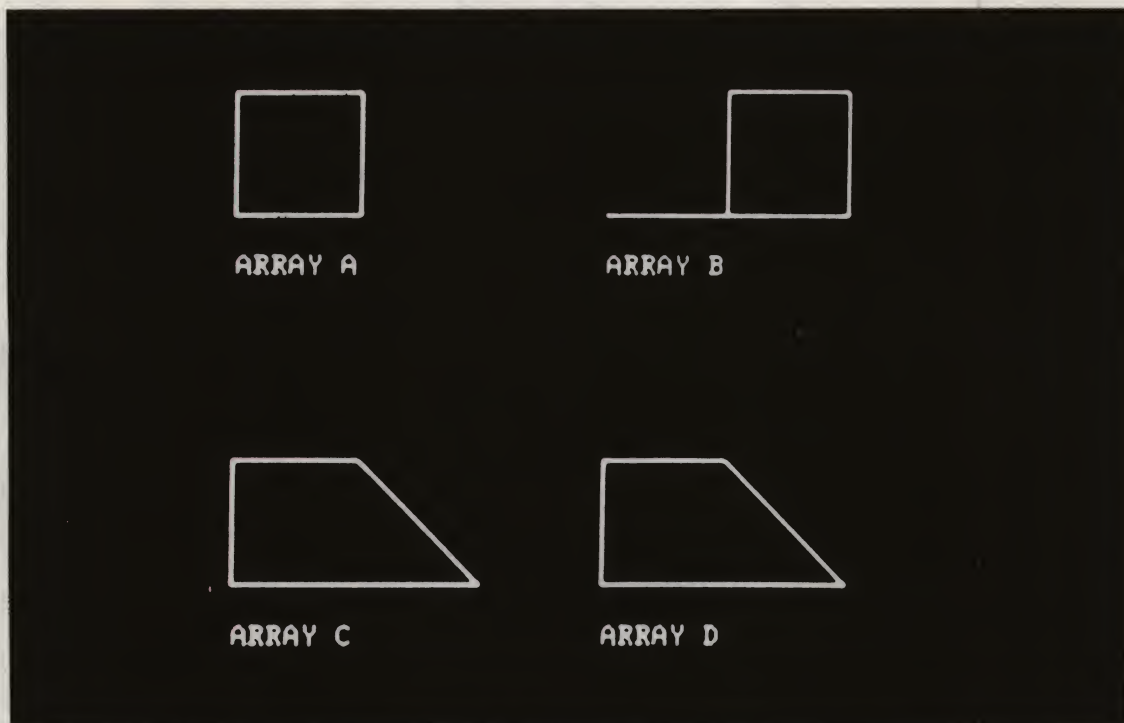
      DIMENSION IBUF(200),A(8),B(8),C(8),D(8)
      DATA A/100.,0.,0.,100.,-100.,0.,0.,-100./
      CALL INIT(IBUF,200)
C     PUT UP THE ARRAY A WITH FIGR
      CALL APNT(100.,500.,0,-5)
      CALL FIGR(A,8)
      CALL APNT(100.,450.,0,-5)
      CALL TEXT('ARRAY A')
C     PUT UP ARRAY B
      CALL APNT(400.,500.,0,-5)
      CALL FIGR(B,8)
      CALL APNT(400.,450.,0,-5)
      CALL TEXT('ARRAY B')
C     PUT UP ARRAY C
      CALL APNT(100.,200.,0,-5)
      CALL FIGR(C,8)
      CALL APNT(100.,150.,0,-5)
      CALL TEXT('ARRAY C')
C     PUT UP ARRAY D

```



## VT11 GRAPHICS SUPPORT

```
CALL APNT(400.,200.,0,-5)
CALL FIGR(D,8)
CALL APNT(400.,150.,0,-5)
CALL TEXT('ARRAY D')
C   EXTEND ARRAY B WITH APUT
CALL APUT(B(1),200.)
C   EXTEND BASE OF ARRAY C WITH APUT'S
CALL APUT(C(1),200.)
CALL APUT(C(3),-100.)
C   DO THE EQUIVALENT TO ARRAY D WITH FPUT
CALL FPUT(D(1),200.)
END
```



Note that figure arrays only use an even number of array elements starting at a subscript of one; the dimension of the array will therefore always be even, as shown in the above example.

All scaling is performed automatically. In the above example, the calls:

```
CALL FPUT(A(7),B)
```

```
CALL FPUT(A(8),B)
```

would have acted exactly like the corresponding APUT calls, because there are no further points (X,Y-pairs) in the array to be adjusted.

### 1.2.9 Using Timing Routines (TIMER and TIMR)

Many graphics routines require the use of the line-frequency clock, especially when pictures must be on the screen for a certain number of seconds and then moved or erased.

**TIMER**

You can set a count-down timer to a value of *nz* ticks by issuing the following call:

```
CALL TIMER(nz)
```

If *nz* is negative, an error message will be printed. If the line current is 60-Hz (cycles per second), there will be 60 ticks per second. Therefore to set the clock to count down for one minute, the following call to **TIMER** would be specified:

```
CALL TIMER(60*60)
```

The timer will operate independently of, as well as compatibly with, any other routine that uses the clock.

**NOTE**

Once the timer has been set, it will continue to count down until it reaches zero. It will continue to count down even if the FORTRAN program is exited and another program is being used. To be safe, because the clock is interrupt-driven, you should finish your program with a:

```
CALL TIMER(0)
```

call to insure that the clock interrupt routine is disabled.

**TIMR**

The timer will count down to zero and then remain zero. Its value at any time may be obtained by issuing the call:

```
CALL TIMR(IE)
```

The current value of the timer will be returned in the variable *IE*.

**NOTE**

If the hardware does not include a line-frequency clock, **TIMR** will always return a value of zero.



## EXAMPLE

Put a subpicture with tag 2 on the screen, and then erase it 15 seconds later.

```

      . . .
      CALL SUBP(1)
C     CREATE SUBPICTURE
      . . .
      CALL ESUB
      CALL TIMER(15*60)
1     CALL TIMR(IE)
      IF (IE.NE.0) GOTO 1
      CALL ERAS(1)
      . . .

```

### 1.2.10 Condensing, Storing, and Retrieving The Display File (SAVE, CMPRS, RSTR)

#### SAVE

After you have executed a portion of a graphics program, the display file may become filled with subpictures that are no longer needed (i.e., they have been erased). A call to SAVE will compact the display file by eliminating the code for erased subpictures and the references to graphic arrays. The data in the graphic arrays will be lost; if you want to save this graphic array data, you must save it before issuing a call to SAVE. The form of the SAVE call is

```
CALL SAVE[(file)]
```

where file is a literal string of the form:

```
'[dev:]filnam[.ext]'
```

If no file is specified, then the display file is condensed, the graphic array references are eliminated, and the program continues. The device can only be a block-replaceable device, such as DECTape or disk. The default device is DK, the system device. The default extension is DPY.

#### CMPRS

You can alternatively issue a call to CMPRS. A CMPRS call with no arguments performs a function similar to the SAVE function, except that CMPRS keeps any currently displayed graph data as part of the display file. It is issued as follows:

```
CALL CMPRS
```



**RSTR**

Saved files may be called in from the disk or DECTape by a call to RSTR. A file may be restored into either an empty buffer or a non-empty buffer. The restored file will be placed at the first free location in the display buffer. The form of the call is:

CALL RSTR (file)

where file is a literal string of the form:

'[dev:]filnam[.ext]'

This feature allows several programs to create pictures, and another program to use these pictures without the overhead of the code required to create them. An application of this feature is the creation of a picture library. Subpictures are created, turned off, and then saved. When the file is restored, those subpictures may be either turned on or copied by means of the two-parameter subpicture calls. This is useful in many applications. For example, a library of electronic components can be built and called in and used when needed without having to actually redraw the components each time the library is viewed.

When the display file contains data and a file is restored at the end of the current data, the tags that are in the original buffer may duplicate those in the restored file. This means that any reference to a given tag will always apply to the first instance of that tag. You should attempt to avoid this situation by carefully numbering subpictures. However, if two subpictures do have the same tag, the original subpicture can be copied with a new tag (in the same location if desired), and then erased. The restored subpicture will then be accessible by its tag, and the original subpicture will be accessible by its new tag.

The display processor treats the restored file as if the code required to create it had been entered at that point. Specifically, after the call to RSTR, all display parameters (l, i, f, t, ns and np) have the values they were assigned in the program that created the restored file.

The monitor USR module must be locked in core if the SAVE/RSTR routine is used.

**EXAMPLE**

Restore two display files as subpictures 1 and 2 respectively. The parameter conditions of the display file should be checked at both the beginning and the end of such files.

```

      . . .
      CALL SUBP(1)
      CALL RSTR('FILE1.DPY')
C     THE NAME COULD HAVE BEEN 'FILE1'
      CALL ESUB
      CALL SUBP(2)
      CALL RSTR('FILE2')
      CALL ESUB
      . . .

```



## CHAPTER 2

### BUILDING AND ALTERING THE GRAPHICS PACKAGE

#### 2.1 BUILDING A LOAD MODULE

The VTll graphics routines are provided as a file of concatenated object modules and as a library:

|                                             |                                                                         |
|---------------------------------------------|-------------------------------------------------------------------------|
| GRAFIX.OBJ                                  | FORTTRAN graphics library                                               |
| VT OBJ.OBJ                                  | FORTTRAN graphics file of concatenated object modules                   |
| VT A.OBJ,VT B.OBJ,...,VT M.OBJ and VT U.OBJ | Individual object files that make up the library                        |
| VT A.OBJ                                    | RDOT, APNT, VECT, and LVECT processor                                   |
| VT B.OBJ                                    | TRAK and LPEN processor                                                 |
| VT C.OBJ                                    | TIMER and TIMR processor                                                |
| VT D.OBJ                                    | NMBR, TEXT, and STAT processor                                          |
| VT E.OBJ                                    | SUBP, ESUB, ERAS, ON, and OFF processor                                 |
| VT F.OBJ                                    | XGRA, YGRA, AGET, and APUT processor                                    |
| VT G.OBJ                                    | SCAL and NOSC processor                                                 |
| VT H.OBJ                                    | INIT, BLNK, UNBLNK, STOP, CONT, FREE, DPTR, DPYWD, and DPYNOP processor |
| VT I.OBJ                                    | FIGR and FPUT processor                                                 |
| VT J.OBJ                                    | SAVE, CMPSR, and RSTR processor                                         |
| VT K.OBJ                                    | CROL processor                                                          |
| VT L.OBJ                                    | General monitor-dependent support routines                              |
| VT M.OBJ                                    | System-dependent code for SAVE and RSTR                                 |
| VT U.OBJ                                    | General utility routines                                                |

The user can merge the VT Handler object file, VTHDLR.OBJ, into the graphics library for convenience, as shown in the example below.

To build the library (VTLIB) from the individual object files, use the following commands

```
.R PIP
*VT OBJ.OBJ=VT A.OBJ,VT B.OBJ,VT C.OBJ,VT D.OBJ,VT E.OBJ,VT F.OBJ
*VT OBJ.OBJ=VT OBJ.OBJ,VT G.OBJ,VT H.OBJ,VT I.OBJ,VT J.OBJ,VT K.OBJ
*VT OBJ.OBJ=VT OBJ.OBJ,VT L.OBJ,VT M.OBJ,VT U.OBJ
*^C

.R LIBR
*GRAFIX=VT OBJ
*VTLIB=GRAFIX,VTHDLR
*^C
```

The following example assumes that the user has a FORTRAN main program named TEST.F4 which uses the graphics support. The USR swaps in normally over the user's data region starting at address 1000. For this reason, if the USR will be used for I/O or possibly if calls are



## BUILDING AND ALTERING THE GRAPHICS PACKAGE

made to the SAVE or RSTR routines, the user's program should be compiled with the U switch to prohibit USR swapping. After successfully compiling TEST, the user should type the following commands

```
.R LINK
*TEST,MAP=TEST,VTLIB/F
*~C
```

The user now has a file named TEST.SAV, which may then be run like any other core-image program.

To assemble the sources VTA.MAC through VTU.MAC, it is necessary to assemble each of these files with the VT parameter file VTMAC.MAC (part of the VT Handler package). An example of this is shown below.

```
.R MACRO
*VTH=VTMAC,VTH
*~C
```

### 2.2 TECHNICAL DESCRIPTION OF DISPLAY FILE

The following technical information should be read before any user-written Assembly Language routines alter the graphic files or before the sources of the graphics support routines are altered.

#### 1. Overall Structure

##### a. Root portion

- i) This is controlled by the VT Handler.

##### b. User space

- i) User data starts at address in global ACTST.
- ii) Data continues through address in global ACTEND.
- iii) A display STOP and zero are stored in the two words ending at ACTEND. The words to be inserted in place of these 2 words are held in WD1 and WD2, respectively.
- iv) The end of the display buffer is the address in global DCRASH.
- v) The DPTR routine returns an integer I such that IBUF(I) points to the display STOP of iii.

#### 2. Subpictures

##### a. Call consists of 5 words:

- i) A display HALT used as a JSR (by software code 173400)
- ii) Address of the rest of the file
- iii) Address of the subpicture
- iv) The tag of the subpicture
- v) The pointer to the next tag or zero if the last subpicture.

##### b. Header of a subpicture consists of one spare word used in the SAVE and CMPRS routines. The address in 2a-ii points to the next word.

##### c. The end of a subpicture is given by 2 words:



## BUILDING AND ALTERING THE GRAPHICS PACKAGE

- i) A display RETURN
- ii) A zero word
- d. Subpictures are turned off and on by interchanging the display JSR of 2a-i by a display JUMP respectively.
- e. When subpictures are erased:
  - i) The display JSR of 2a-i is replaced by a display JUMP.
  - ii) The tag of the subpicture and the tags of any sub-pictures contained in it are deleted from the linked list.
- f. The first subpicture tag pointer is located at address global TAG1+2.
- 3. Display stops are serviced by the VT Handler.
- 4. Light pen hits are serviced by the VT Handler.
- 5. Display time-outs are serviced by the VT Handler.
- 6. Calls to XGRA, YGRA, FIGR.
  - a. The following code is generated for XGRA and YGRA:
    - i) Set Graphic Mode to XGRA or YGRA.
    - ii) Load Status Register B with the graphplot increment.
    - iii) A display JUMP to the first word of A(i).
    - iv) Address of the first word of A(1) where the screen-scaled array data starts. The data will be compressed into one word entries suitable for the display processor.
    - v) A key indicating the type of array; 0=XGRA, 2=YGRA, and 4=FIGR.
  - b. The display JUMP at the "end" of the array points to the word following 6a-v.
  - c. For FIGR. 6a-i and 6a-ii are replaced by a single word to set the graphic mode to long vector.
- 7. Calls to NMBR routine
  - a. These calls are similar to the sequence SUBP, TEXT, and ESUB, except that the text graphic mode word has bit 0 set and it is followed by 16 bytes for data.
- 8. File structure as a "Saved File"
  - a. The saved file has no references to "graph" arrays because restoring the file into a program without such arrays is meaningless.
  - b. The file has a total of n+2 words where the first word is n and the second word is a relative offset from the next word to the first subpicture tag if there is one; else it is zero.

## BUILDING AND ALTERING THE GRAPHICS PACKAGE

- c. The file is stored entirely in relocatable form relative to the third word of the file. Any restoration which you attempt with your own routines may result in "garbage" unless care is taken to locate all the addresses properly.
9. Display buffer organization on sequential calls to the same module (vector/point/graph plot module).

The example provided below makes sequential calls to the module LVECT.

A single call to LVECT is normally mapped into three words in the display buffer.

```
word.1 Mode M
word.2 X coordinate X
word.3 Y coordinate Y
```

But on a sequential call to LVECT as indicated below

```
...
CALL LVECT(X1,Y1)
CALL LVECT(X2,Y2)
CALL LVECT(X3,Y3)
..
```

these calls will be mapped into the display buffer as shown below

```
...
M
X1
Y1
X2
Y2
X3
Y3
..
```

This is because the mode word (l,i,f,t) remains unaltered in the second and third call.



## INDEX

- Absolute point, 1-13
- Array,
  - figure, 1-24, 1-25, 1-26, 1-27
  - graphic, 1-24, 1-25, 1-26
- BASIC, 1-24
- Beam, 1-14, 1-15, 1-22, 1-24, 1-25
- Blink mode, 1-14, 1-15
- BUILDING LOAD MODULE, 2-1
- Carriage return, 1-16
- Characters,
  - italic, 1-18, 1-19
- Clock,
  - line-frequency, 1-2, 1-27, 1-28
- Compressing display file, 1-29
- Concluding subpictures, 1-20
- Coordinate system, 1-12
- Count-down timer, 1-28
- Defining subpictures, 1-19, 1-20
- Display file, 1-3
  - compressing, 1-29
  - restoring, 1-30
  - saving, 1-29
- Display file accessing
  - words, 1-10, 1-11
- Display file compressing
  - space, 1-8
- Display file internal
  - structure, 2-2
- Display screen, 1-2, 1-3
- Display screen clearing, 1-9
- Display screen scaling, 1-12
- Display screen turning off, 1-9, 1-10
- Display screen turning on, 1-9, 1-10
- Displaying text, 1-16
- Distribution media, 1-1
- Erasing subpictures, 1-21
- Figure array, 1-24, 1-25, 1-26, 1-27
- Flash mode, 1-14, 1-15
- FORTTRAN compiler optimizing code, 1-23
- Graphic array, 1-24, 1-25, 1-26
- Initializing the system, 1-9
- Intensity, 1-14, 1-15, 1-19
- Interrupt,
  - Light pen, 1-14
- INTSET call, 1-8
- Invisible words, 1-18
- Italic characters, 1-18, 1-19
- Light pen, 1-14, 1-15, 1-22, 1-23
- Light pen hit, 1-18, 1-19, 1-22
- Light pen interrupt, 1-14
- Line feeds, 1-16
- Line type, 1-14, 1-15
- Line-frequency clock, 1-2, 1-27, 1-28
- LOAD MODULE,
  - BUILDING, 2-1
- Long vector, 1-10, 1-15
- MODULE,
  - BUILDING LOAD, 2-1
- Nested subpictures, 1-20, 1-21
- Null byte, 1-18, 1-22
- Numeric data positioning on screen, 1-22
- Numeric subpicture, 1-22

## INDEX (CONT.)

Point,  
    absolute, 1-13  
    relative, 1-15  
.PROTECT request, 1-8

Relative point, 1-15  
Restoring display file,  
    1-30

Saving display file, 1-29  
Scaling, 1-14  
Scroller, 1-10, 1-19  
Sensitivity light pen, 1-14  
Shift-out characters, 1-16  
Short vector, 1-15  
Subpicture,  
    numeric, 1-22  
Subpicture tag, 1-20, 1-21,  
    1-22, 1-30  
Subpictures, 1-19, 1-20,  
    1-21, 1-22  
Subpictures,  
    concluding, 1-20  
    defining, 1-19, 1-20  
    erasing, 1-21  
    nested, 1-20, 1-21  
    turning off, 1-21  
    turning on, 1-21

Tag,  
    subpicture, 1-20, 1-21,  
        1-22, 1-30  
Text,  
    displaying, 1-16  
Timer,  
    count-down, 1-28  
Tracking object, 1-10, 1-23  
Truncation of vectors, 1-2  
Turning off subpictures,  
    1-21  
Turning on subpictures,  
    1-21

USEREX routine, 1-9

Vector, 1-15  
    long, 1-10, 1-15  
    short, 1-15  
VT11 display processor, 1-1  
VT11 handler, 1-2, 2-1



READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Performance Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

---

---

---

---

---

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

---

---

---

---

---

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

---

---

---

---

---

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_

Street \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_  
or  
Country

If you require a written reply, please check here.

Please cut along this line.

-----Fold Here-----

-----Do Not Tear - Fold Here and Staple-----

FIRST CLASS  
PERMIT NO. 33  
MAYNARD, MASS.

BUSINESS REPLY MAIL  
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

**digital**

Software Communications  
P. O. Box F  
Maynard, Massachusetts 01754

